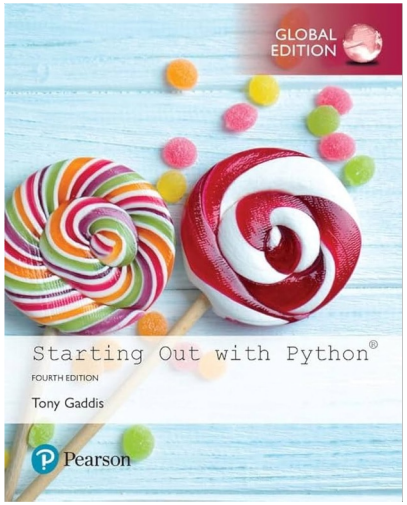
Data Types, Expressions and Variables

CIIC 3015 - Lecture 1

Jean Carlos Méndez
With some updates by: Bienvenido Vélez

Lecture Goals

- **Data Types**
  - What are they?
  - Demo
- **Operations**
  - Numerical Operations
  - String Operations
  - Demo
- **Variables**
  - What are they?
  - How can they be used?
  - Demo



Chapter 2
Input, Processing and Output

Data Types

Data Types

- Data itself is a fundamental building block of Computer Science and programming.
- Each data “value” has its own type, i.e. numbers, words, images etc.
- These data types each have their own properties that make them unique.
- Python provides specific syntax for expressing values of each distinct data type

Data Types: **integer**

integer: number without a decimal point

- Abbreviated as **int**

example values

```
5

-40

0

13382
```

Google | GH |

Data Types: **float**

float: number with a decimal point and a fractional value

- Comes from term ***floating point number***

example values

```
3.14

2.0

-9.5
```

Data Types: **integer** vs. **float**

- It is faster to do math with **ints** than with **floats**
- **ints** occupy less space in memory
- **floats** can represent numbers that **ints** cannot
- Some cases are **int**-specific and vice-versa
  - can you be in 2.35th place? (**int** better)
  - how to represent π more closely? 3 vs. 3.14 (**float** better)
- In general, prefer **ints** over **floats** unless you *need* **floats**
  - Hint: You will *need* **floats** **very** often

Data Types: **string**

string: sequence of characters with single quotes or double quotes

- In other words, it's just any text between quotes
- Abbreviated as **str**
- Both single quotes (‘’) and double quotes (“”) work
  - Choose one and stick with it!

example values

```
'abcdefg'

“Hello!”

'3.14'
```

Data Types: **booleans**

boolean: either **True** or **False**

- Abbreviated as **bool.**

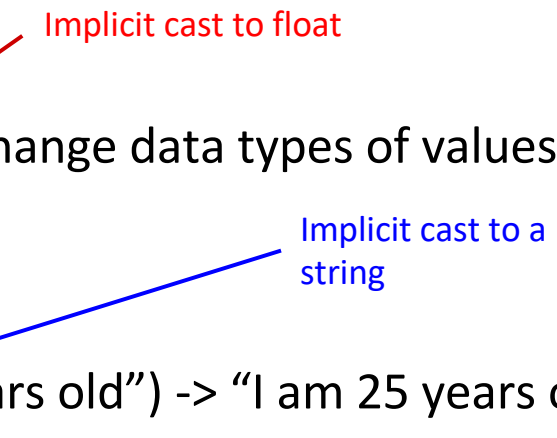
Note: Most languages use lowercase (true and false), but they are capitalized in Python

example values

```
True

False
```

Data Types: Casting

- Changing the data type of a value
- Explicit Casting - Changing data types of a value by using float(), bool(), int(), str() functions
- Ex: float(5) -> 5.0
- Ex: int("5") -> 5
- Implicit Casting - Computer change data types of values to evaluate the expression
  - Ex: 25 + 2.5 -> 25.5
  - Ex: print("I am", 25 , "years old") -> "I am 25 years old"



Data Types and Input/Output: Built-in functions

int() converts input value to an integer

float() converts input value to a float

str() converts input value to a string

bool() converts input value to a boolean

type() determines data type of input value

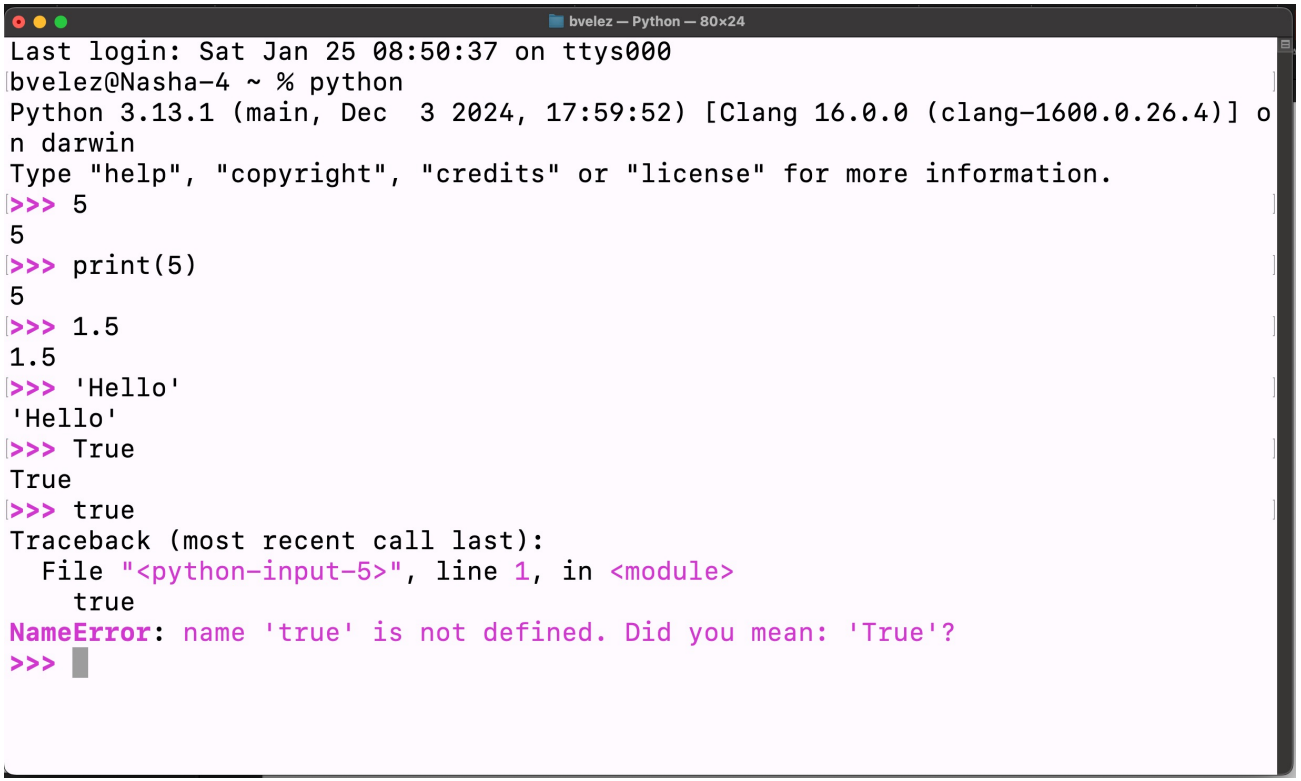
print() displays input value in console

input() receives input from the user in console

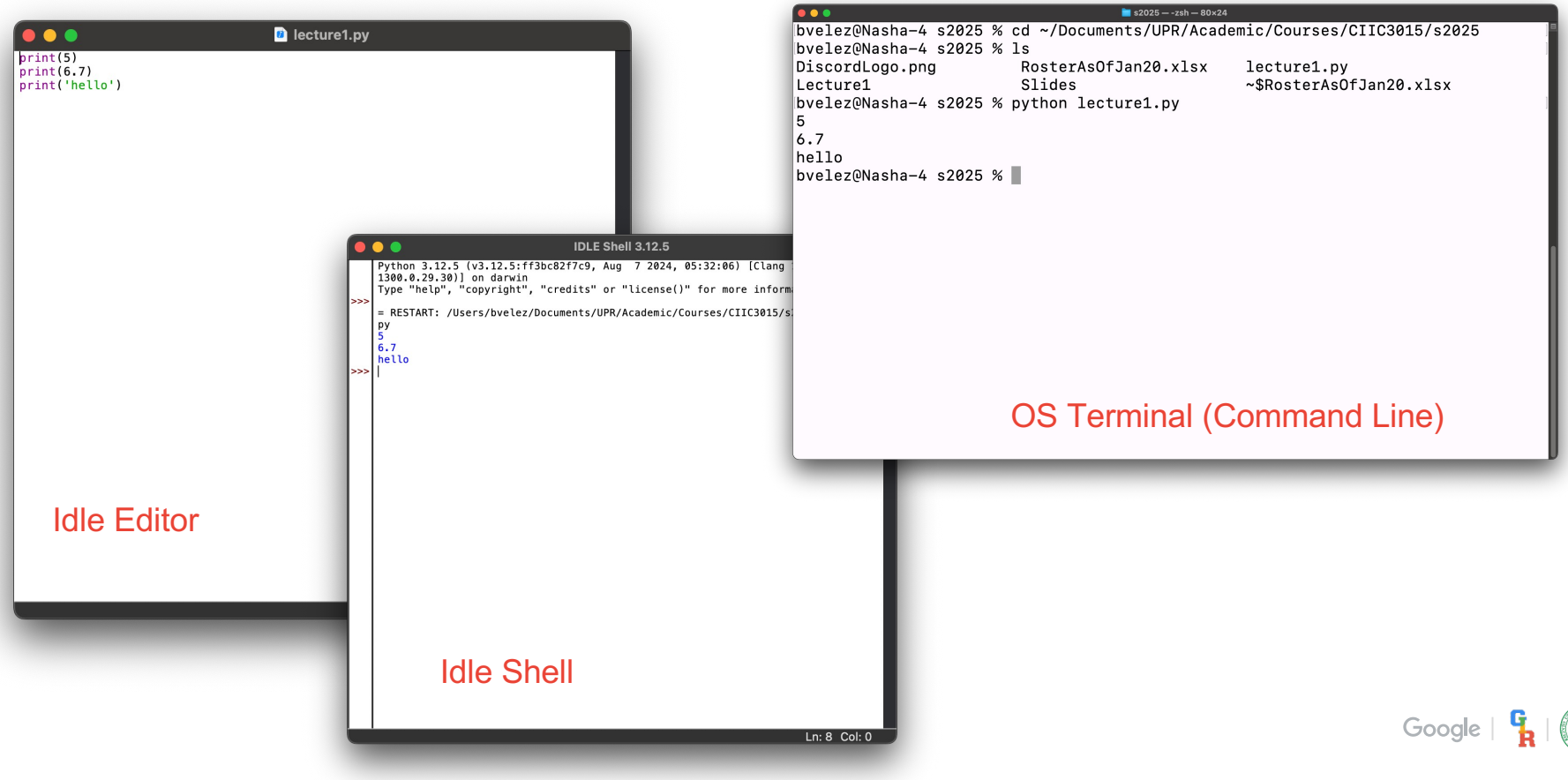
Three Main Modalities to Run Python Code:

- Interactive Mode (a.k.a. command line)
- Scripting Mode
- Integrated Development Environment (IDE) Mode

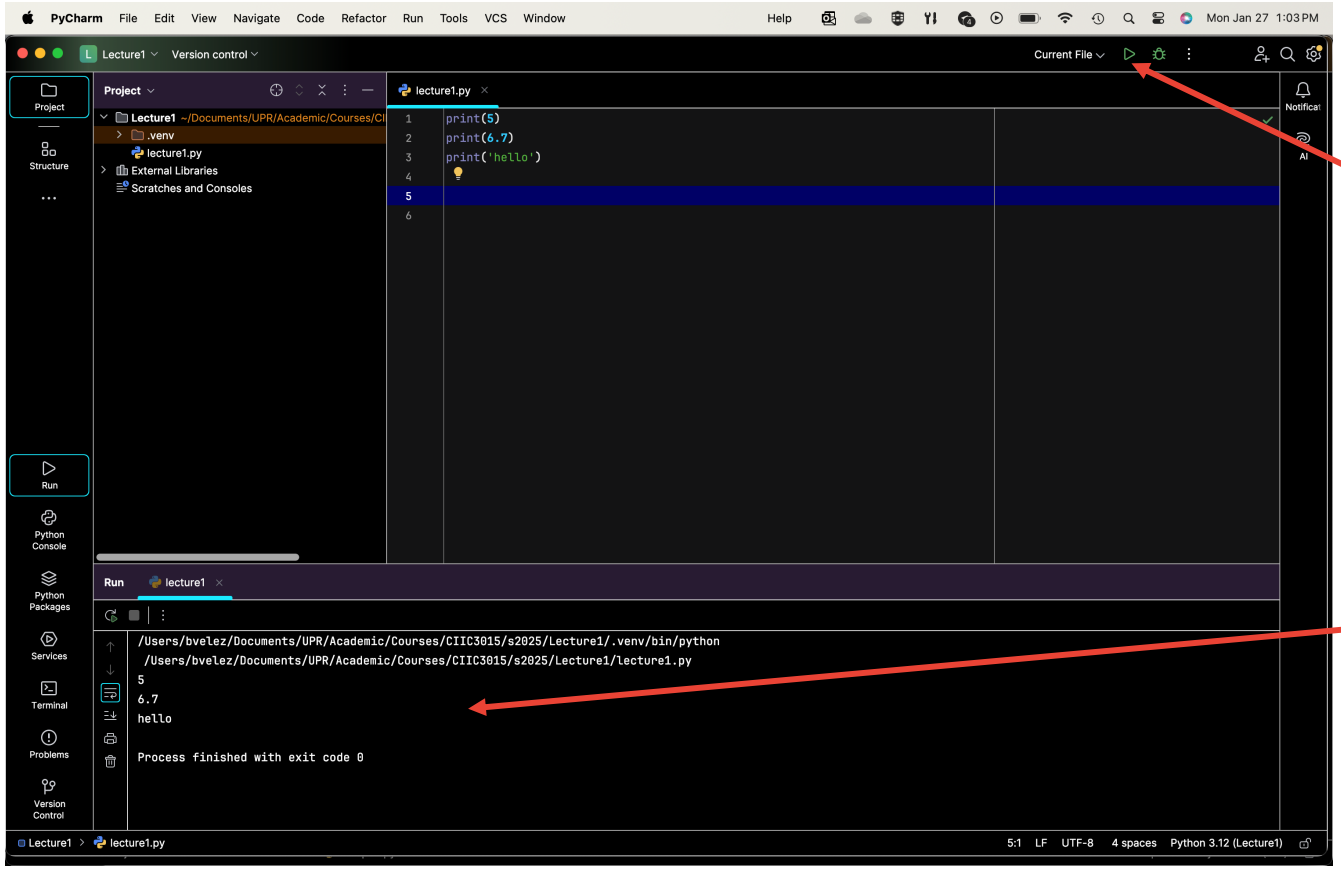
Running Code in Interactive Mode



Running Code in Scripting Mode (Idle)



Running Code in IDE Mode (Pycharm)



Run Button

Console Output

Live Coding! Example 1

- Print
- Comments
- Data Types
- Casting

Code available in:
github.com/CSwithJC/CIIC3015/blob/main/lecture_01/example_01.py

Data Types: Summary

int	Number with **no** decimal point	`5, -17, 0, 2874976`
float	Number with decimal point	`2.5, 3.00, -12.95, 7.`
str	Text between quotes	`'Hello', "", "False", '53.12'`
bool	True or False	`True, False`

Data Types practice: **int**, **float**, **str**, or **bool**?

`"hi, mom"` **str**

`2.71828` **float**

`1000000` **int**

`true` **not a value!**

`'1,000,000'` **str**

`False` **bool**

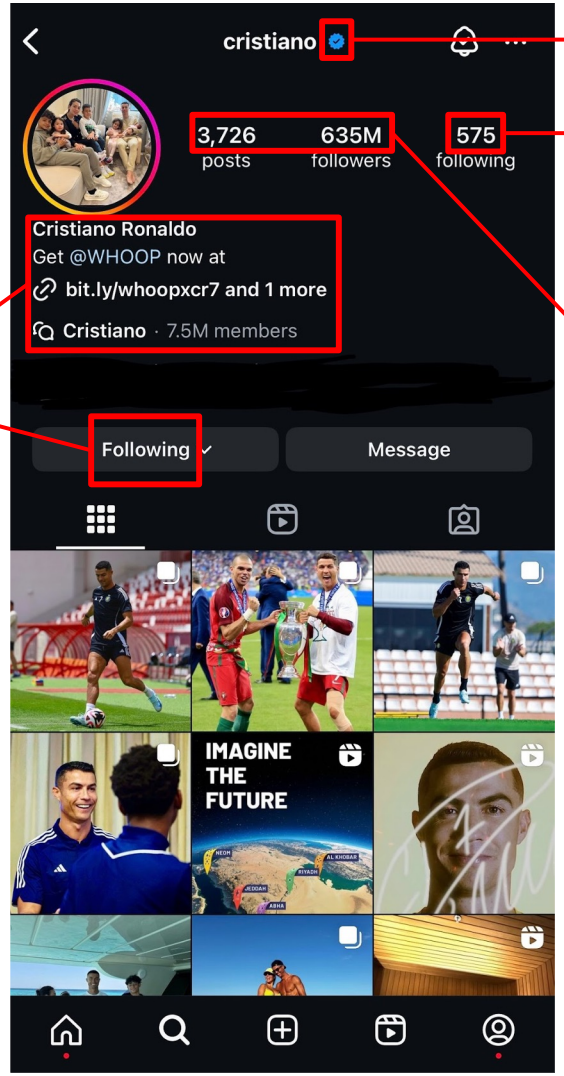
`-0.0000009` **float**

`squirrel` **not a value!**

Data Types practice: **int**, **float**, **str**, or **bool**?

`type("hi, mom")`	<class `str`>	`type('1,000,000)'`	<class `str`>
`type(2.71828)`	<class `float`>	`type(False)`	<class `bool`>
`type(1000000)`	<class `int`>	`type(-0.00009)`	<class `float`>
`type(true)`	true undefined!	`type(squirrel)`	squirrel undefined!

Data Types: Instagram Example



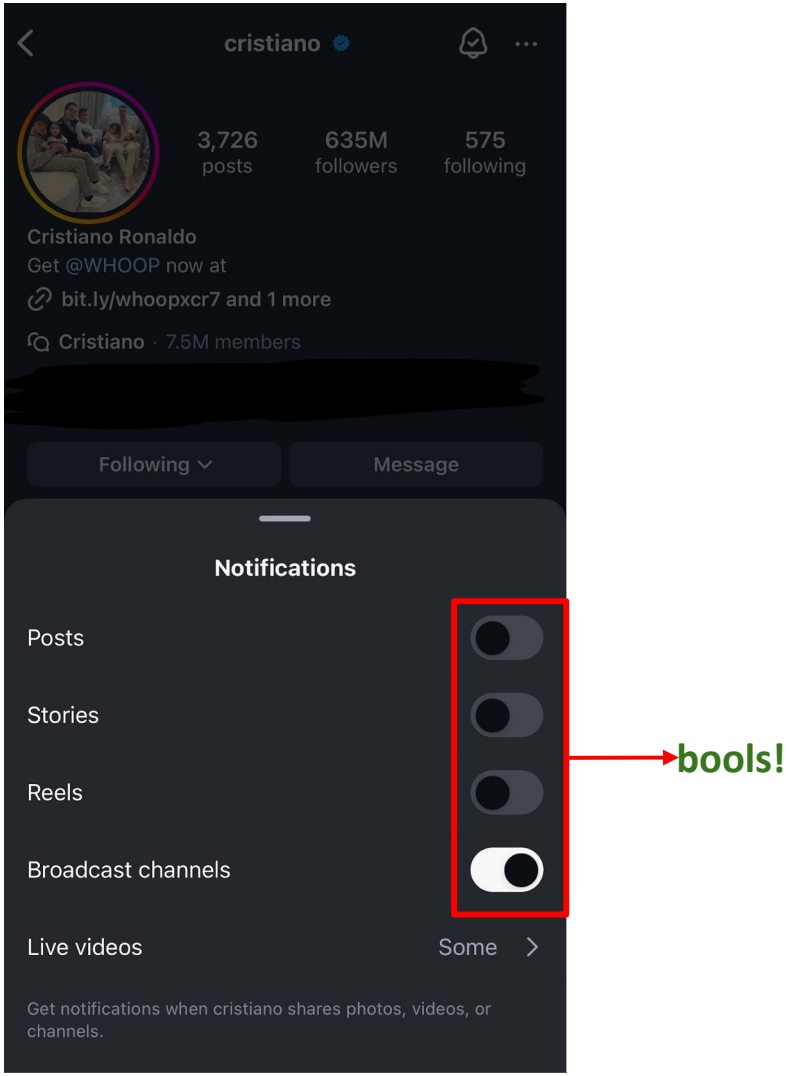
Shown based on a **bool**

int

str (but derived from **int**)

str

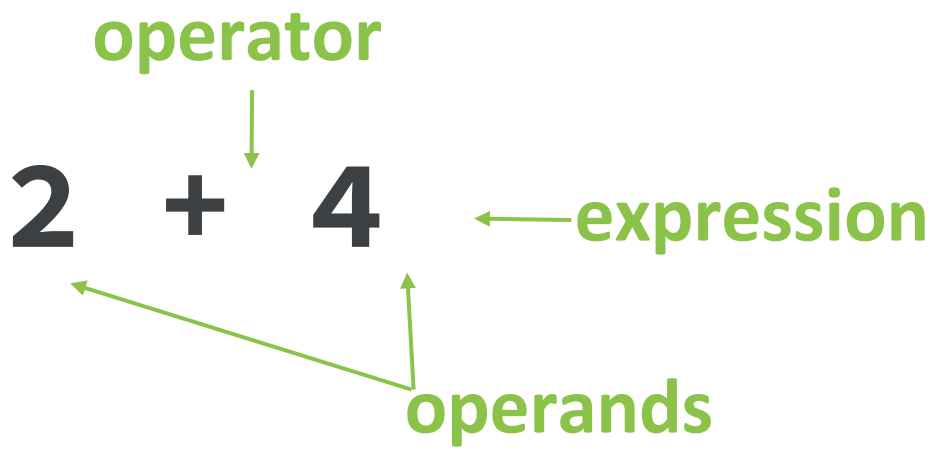
Data Types: Instagram Example



Operators

Operators

- **operator**: symbol performing specific computation on set of values
- **expression**: combination of operators and values evaluating to single value



Google |

Arithmetic Operators

+	Add	//	Floor Division
-	Subtract	%	Modulo
*	Multiply		
/	Divide		
**	Exponent		

Arithmetic Operators: Basic Examples

3 + 4	=	7
3 - 4	=	-1
4 * 2	=	8
4 / 2	=	2.0

Arithmetic Operators: Floats, Integers

- If **any** operand is **float**, evaluated value is **float**
  - 1 + 2.0 = 3.0 (not 3)
- If **all** operands are **integers**, evaluated value is **integer**
  - 4 * 3 = 12 (not 12.0)
- **EXCEPT DIVIDE** / - evaluated value is **always float**
  - 12 / 4 = 3.0
  - 12.0 / 4.0 = 3.0
  - 12.0 / 4 = 3.0

Arithmetic Operators: Strings

\+ Add

"And" + "rea" => "Andrea"

Note: Adding strings together is called **concatenation**

* Multiply

"hello" * 2 => "hellohello"

"hello" * 0 => ""

"hello" * -1 => ""

Note: string multiplication only exists in Python

Arithmetic Operators: String Expressions Practice

```
"Hi" + "mom"    =   "Himom"

"lol" + "ha" * 3 = "lol" + "hahaha" = "lolhahaha"

("lol" + " ha") * 3 = "lol ha" * 3 = "lol halol halol ha"

"b" * 2 + 8     =   "bb" + 8     =   TypeError: can
                                     only concatenate
                                     str (not "int") to
                                     str
```

Google

Arithmetic Operators: String Expressions Practice

```
"b" * 2 + 8    =  'bb' + 8    =  TypeError: can only concatenate str (not "int") to str
```

```
"b" * 2 + "8"  =  "bb" + "8"  =  "bb8"
```

Arithmetic Operators: Order of Operations

Parentheses
Exponents
Multiplication
Division
Addition
Subtraction

- MD and AS have the same precedence, go left to right
- MD, //, and % have the same precedence, go left to right

Arithmetic Operators: PEMDAS

```
2 + 1 * 5 - 6 / 2   =   6 - 5 + 3   =   1 + 3 = 4

3 + (1 + 4) ** 2 =  3 + 5 ** 2 =   3 + 25 = 28

"a" + "b" * 2 + "c" =  "a" + "bb" + "c" = "abbc"
```

Arithmetic Operators: Exponentiation (**)

- Multiply a value a certain number of times by itself
- How do people refer to this operator?
  - 4**2 → “four to the power of two”

```
4**2 = 4*4 = 16

2**4 = 2*2*2*2 = 16

3**1 = 3
```

Google | GH |

Arithmetic Operators: Floor Divison (//)

- Divides two numbers and rounds result down to nearest **smaller** whole number
  - When the result is positive, it is truncated, which means that its fractional part is thrown away.
  - When the result is negative, it is rounded away from zero to the nearest integer.

```
5 // 2       =  2      <-- 5 / 2 is 2.5, so remove the 0.5
5.0 // 2.0   =  2.0

42 // 10     =  4      <-- 42 / 10 is 4.2, so remove the 0.2

9.0 // 9.0   =  1.0    <-- 9 / 9 is 1, so nothing changes

-5 // 2      =  -3     <-- -5 / 2 is -2.5, so goes down to -3
```

Google | GH |

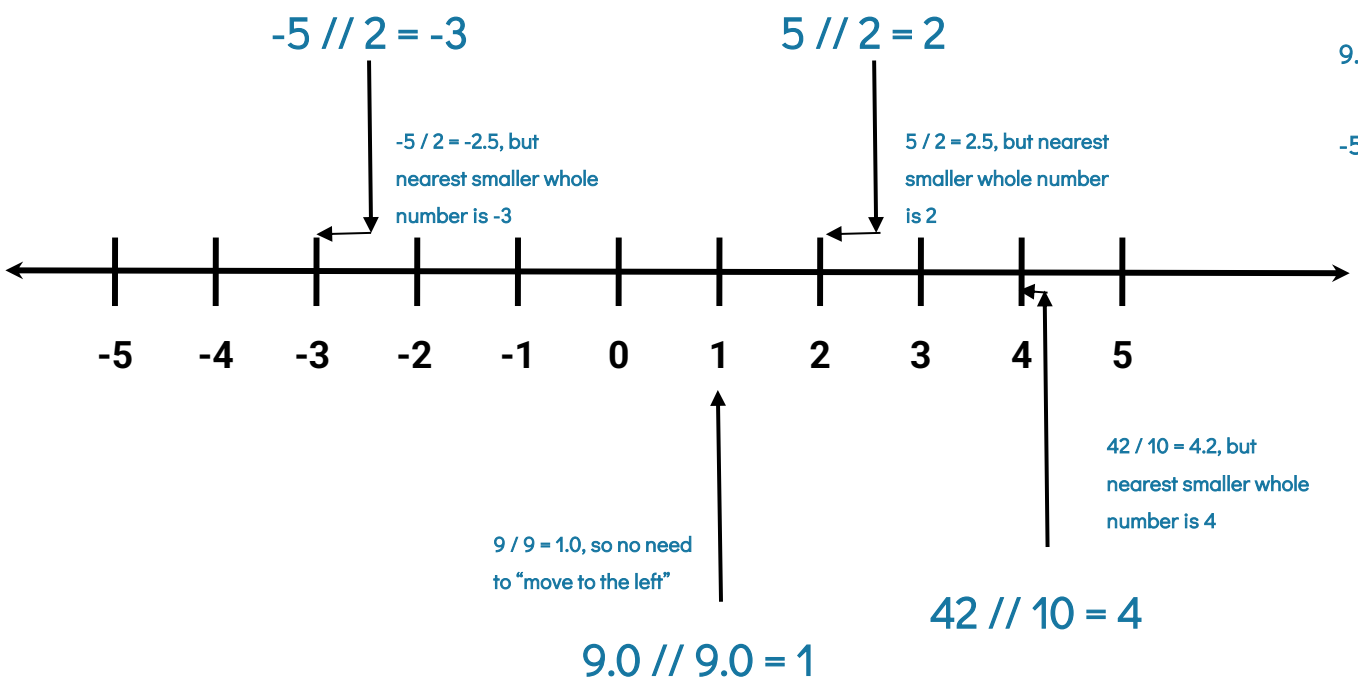
Arithmetic Operators: Floor Divison (//)

5 // 2

42 // 10

9.0 // 9.0

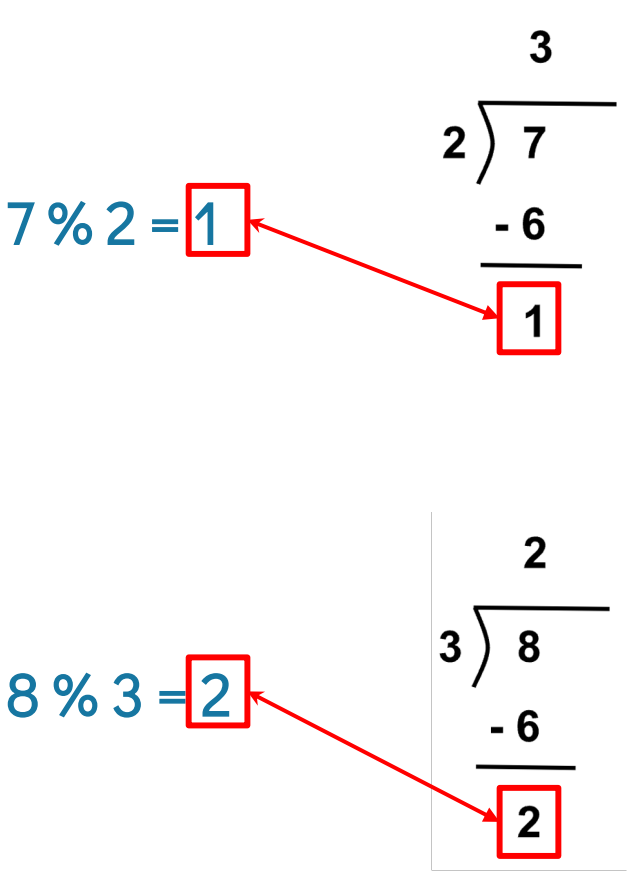
-5 // 2


-5 // 2 = -3
5 // 2 = 2
-5 / 2 = -2.5, but nearest smaller whole number is -3
5 / 2 = 2.5, but nearest smaller whole number is 2
-5 -4 -3 -2 -1 0 1 2 3 4 5
42 / 10 = 4.2, but nearest smaller whole number is 4
9 / 9 = 1.0, so no need to "move to the left"
42 // 10 = 4
9.0 // 9.0 = 1

Pro tip: Do a normal division between both numbers. If there is the division has a fractional part, then move to the nearest int to the LEFT in a number line

Arithmetic Operators: Modulo (%)

- Divide, and take the **remainder**
- How do people refer to this operator?
  - 7 % 2 → “seven mod 2”
- What do you think would happen if we try to mod by 0? (e.g. 8 % 0)



Operators and Expressions: Practice

What is the Value and Datatype of Each Expression?

`0.0 + 1` **1.0 (float)**

`int('5' + '1') + 3` **54 (int)**

`str(4.9) * 3` **'4.94.94.9' (str)**

`type('3.1' + "4")` **<class 'str'>**

`'cat' * -3` **'' (str)**

`'cat' * -3.0` **TypeError**

`float(4)` **4.0**

`int(-4.8)` **-4**

`bool(0)` **False**

`float('banana')` **ValueError: could not convert string to float: 'banana'**

Questions?

Post to our Discord server
or
email: upr-ciic3015-staff@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

Google | GH |

Variables

Variables

A **variable** is a named piece of memory where the programmer can store and retrieve a value.

```
num_de_municipios = 78
tax_rate = 0.115
my_name = 'jean carlos'
i_love_cs = True
```

Variables

We say a variable is **defined** the first time it is assigned (given) a value.

```
num_de_municipios = 78
```

```
tax_rate = 0.115
my_name = 'jean carlos'
i_love_cs = True
```

Variable Naming

- Letters a-z, upper / lowercase, numbers 0-9, underscore (_)
- Can't begin with number
- Cannot contain operator
- Case sensitive
- No spaces, separate words with underscore
- No reserved words (technically works but will break code)

Good “Style”

- Use lowercase letters only
- Begin with lowercase letter
- Use underscores between words

E.g. good_style = 100

Variables: Do Not Use Python Reserved Words as Variable Names

False	None	True	and
as	assert	async	await
break	class	continue	def
del	elif	else	except
finally	for	from	global
if	import	in	is
lambda	match	nonlocal	not
or	pass	raise	return
try	while	with	yield

Practice: Variable Naming

`my_num` ✅

`1st_num` ❌

`a*b` ❌

`x - y` ❌

`num_1` ✅

`_my_variable` ✅

Good naming is important!!!!!!!!

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

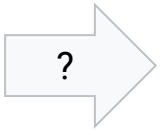
What is this code doing?

```
hours = 35.0
rate = 12.50
pay = hours * rate
print(pay)
```

Variables: Examples (contd.)

What's the final value of each variable?

```
first_num = 10

second_num = first_num * 2

first_num = first_num + 2
```

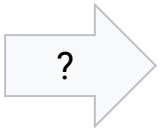



```
first_num = 12

second_num = 20
```

Variables: Examples (contd.)

What's the final value of each variable?

```
a = 5.5

b = 100

temp = a

a = b

b = temp
```

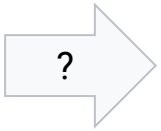



```
a = 100

b = 5.5

temp = 5.5
```

Variables: Examples

What's the final value of each variable?

```
keshav = 18

next_num = 9

next_num = 'goodbye'

rachel = 'purple' + 'green'

Next_num = 'hello' + next_num

keshav = keshav + 1.2

word = rachel
```




```
keshav = 19.2

next_num = ‘goodbye’

rachel = ‘purplegreen'

Next_num = ‘hellogoodbye’

word = ‘purplegreen’
```

Variables: Augmented Assignment Operators

It is very common to update existing variables using arithmetic operators (+, -, *, /, etc.) Due to this, Python provides **augmented assignment operators** for convenience.

Operator	Example Usage	Equivalent To
+=	x += 5	x = x + 5
-=	y -= 2	y = y - 2
*=	z *= 10	z = z * 10
/=	a /= b	a = a / b
%=	c %= 3	c = c % 3

Augmented Assignment Operators: Practice

```
a = 15
b = 20
a += b
```

a =**35**

```
a = 10
b = 5
a *= b
```

a =**50**

```
a = 107
b = 100
a -= b
```

a =**7**

```
a = 10
b = 5
a /= b
```

a = **2.0**

Google | GH |

Variables: input() function

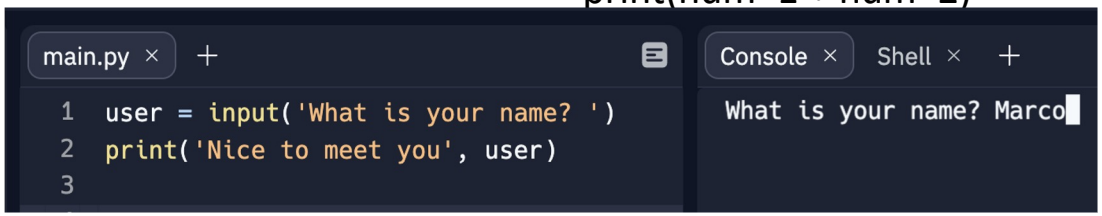
- Helpful function that prompts the user to type input in the console and saves it in a variable.
- The input is always a **string**, so it might need to be casted to a different type, depending on what you want to do with the input.

 # Print your name, typed by you

```
name = input("What is your name?")
print("Hello, " + name + "!")
```

 # Add two numbers you type yourself

```
num_1 = int(input("First number you want to add?"))
num_2 = int(input("Second number you want to add?"))
print(num_1 + num_2)
```



Live Coding! Examples 3

- Unit Conversions (e.g. Fahrenheit <-> Celsius)
- Perimeter of a rectangle given its sides
- Area of a circle given its radius
- Distance between two points given their coordinates (x, y)
- Savings account 1 year earnings at simple annual interest
- Savings account n year earnings at simple annual interest
- Quadratic equation to find zeros of a polynomial ax^2 + bx + c
- Area of triangle given lengths of its three sides (Heron's formula)

Code available in:
https://github.com/UPRM-CIIC3015-F25/LiveCodingExamples

Questions?

Post to our Discord server
or
email: upr-ciic3015-staff@upr.edu

Give us anonymous feedback!

https://bit.ly/CIIC3015_F25_Feedback

Google | GH |